**ENGR 102 Sect 508 Lab 9b**

**100 points**

**Reading assignment:**

|  |  |
| --- | --- |
| **Lecture Slides** | **L09** |
| **zyBook chapter 9** | **Complete all participation and challenge activities** |

*Attention!!*

*Individual submission.*

*Submit* *your Py-files together with your word/pdf file with screenshots of your tests outputs. Include any derivations, comments and supplemental notes in your word/pdf files.*

*No pictures by the phone – it is impossible to read. You will be allowed to resubmit and reupload HW as many times as you want to within the due date/time, only last submission will be graded. No late submissions. For submission you may use this file as template: rename file including your name. Do not forget to put your name inside of this file as well. If it is a team work, include the team number and all team members. For this submission use Individual Header.*

These activities are meant to help give you practice reading and writing files, as well as processing larger amounts of data, which is one of the common tasks that computer programs are written for.

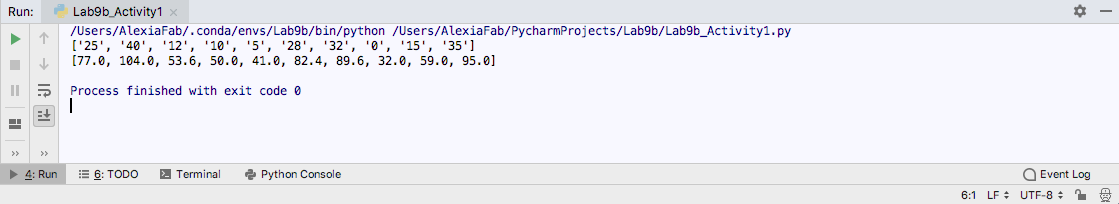
**Activity #1: [15 points]**

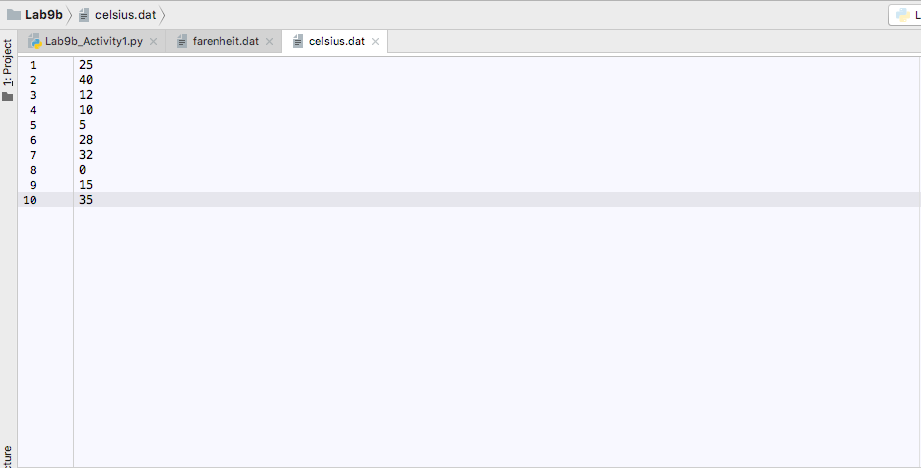
Write a program that will take a file named Celsius.dat that contains a list of temperatures in Celsius (one per line), and will create a file Fahrenheit.dat that contains the same temperatures (one per line, in the same order) in Fahrenheit.

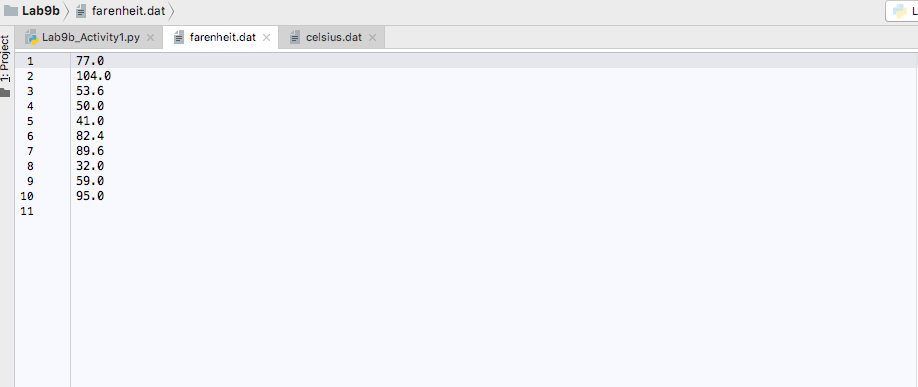
Code:

*# By submitting this assignment, I agree to the following:  
# “Aggies do not lie, cheat, or steal, or tolerate those who do”  
# “I have not given or received any unauthorized aid on this assignment”  
#  
# Name: ALEXIA PEREZ  
# Section: 508  
# Assignment: LAB 9B  
# Date: 28-10-2018***from** math **import \*  
import** numpy  
  
*# This first section opens the file with all the data, reads it and puts it into a string,  
# and then converts the string into a list for easier manipulation:*celsius **=** open(**'celsius.dat'**,**'r'**)  
celsius\_string **=** celsius.read()  
celsius\_list **=** [y **for** y **in** (x.strip() **for** x **in** celsius\_string.splitlines()) **if** y]  
print(celsius\_list)  
  
*# This part of the code convert each element to farenheit and adds them to a new list:*farenheit\_list **=** []  
length **=** len(celsius\_list)  
  
**for** i **in** range (length)**:** number **=** int(celsius\_list[i])  
 farenheit **=** 9**/**5 **\*** number **+** 32  
 farenheit\_list.append(farenheit)  
  
*# Finally we add the values in that list to a new file:*farenheit\_file **=** open(**'farenheit.dat'**,**'w'**)  
**for** i **in** range (len(farenheit\_list))**:** temperature **=** str(farenheit\_list[i])  
 farenheit\_file.write(temperature**+'**\n**'**)  
  
print(farenheit\_list)

Output:







**Activity #2: [35 points]**

One of the most common ways that data can be stored to be loaded in a spreadsheet or other similar table is with a CSV (Comma Separated Value) file. These often are given a .csv extension. A csv file is a way of representing a table in a file. Each line represents a row of the table, and the cells in each column are separated by commas. CSV files can usually be read into spreadsheet programs (such as Excel), and most spreadsheets can output their data in a CSV format (sometimes called “comma delimited” format). You are going to practice writing and reading CSV data directly.

Write a program that will save, to a file, a list of amortized values for a loan. Specifically:

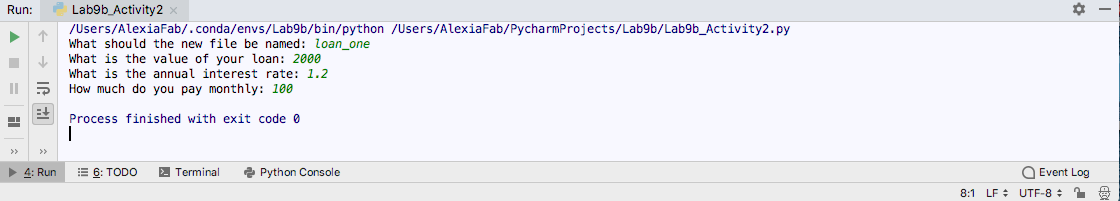
* 1. Ask the user for the amount of the loan, the annual interest rate, and the amount being paid monthly. Also, ask for the name of the file to write the results to. For whatever name is given, your program should add a “.csv” extension to the name, to indicate that it is a CSV file.
  2. Each month, you should calculate the amount remaining on the loan by first applying the monthly payment (reducing the loan value), then increasing the loan amount by 1/12 of the annual interest rate.
  3. For each month, write to the output file the month number, the total amount of interest accrued so far, and the amount remaining on the loan, separated by commas.
     1. Start with month 0, when there is no payment and no interest, with month 1 being the first payment and first interest accumulation
     2. If the loan eventually will be paid off (i.e. if the loan amount is decreasing), write out values until the loan amount is 0
     3. If it will not be paid off (i.e. the loan amount increases or stays the same each month), then write 30 months worth of data.
  4. Be sure that you write out column headers for the table, indicating what each column is.

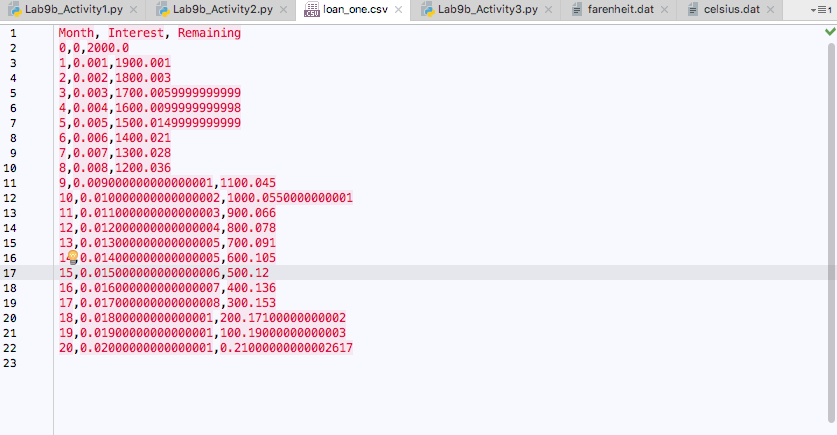
Note: if you write your .csv file correctly, you should be able to open it in a spreadsheet program that can read .csv files.

Code:

*# By submitting this assignment, I agree to the following:  
# “Aggies do not lie, cheat, or steal, or tolerate those who do”  
# “I have not given or received any unauthorized aid on this assignment”  
#  
# Name: ALEXIA PEREZ  
# Section: 508  
# Assignment: LAB 9B  
# Date: 28-10-2018***from** math **import \*  
import** numpy  
  
*# Variable initialization*NewFile\_name **=** str(input(**"What should the new file be named: "**))  
NewFile\_name **+= '.csv'**loan **=** float(input(**"What is the value of your loan: "**))  
interest **=** float(input(**"What is the annual interest rate: "**))  
monthly\_payment **=** float(input(**"How much do you pay monthly: "**))  
remaining **=** loan **-** monthly\_payment  
remaining\_Winterest **=** loan  
total\_remainder **=** 0  
count **=** 0  
new\_file **=** open(NewFile\_name,**'w'**)  
new\_file.write(**"Month, Interest, Remaining"+"**\n**"**)  
  
*# Operations***while** count**<=** 30 **and** loan**>=** 0**:  
 if not**(loan**<=**0) **or not**(count**>=**30)**:** new\_file.write(str(count)**+","+**str(total\_remainder)**+","+**str(loan)**+"**\n**"**)  
 total\_remainder **+=** (interest**/**100) **\*** (1**/**12)  
 remaining\_Winterest **=** (loan**-**monthly\_payment)**+**total\_remainder  
 loan **=** remaining\_Winterest  
 count **+=** 1  
*# closing of the final file*new\_file.close()

Output:

****

****

**Activity #3: [50 points]**

On the class website is a CSV file containing weather data from Coulter Field (in Bryan) for 3 years (1 day is missing for some reason!); the data was taken from Weather Underground (wunderground.com). There are different versions of the file for Windows and Mac; the only difference is whether the end of a line contains just a new-line character, or both a new-line and a carriage return (you don’t need to worry about the difference). Open the file in any text browser and you should see what it is. Note that the first line of the file contains the column headers explaining what each column is.

Download the file to your system, and write a program that will read in the file and do the following:

1. Output the maximum and minimum temperature seen over the 3 year period
2. Output the average daily precipitation
3. Pick any 3 other “interesting” data analysis questions (of your choice) and output the answer to that question. For at least one, make use of the date information. Here are some ideas, but you can pick whatever you want:
   1. For some particular day, such as December 25, find the maximum and minimum temperatures reached among the 3 years of data.
   2. For some particular month, such as July 2015, calculate the average high temperature.
   3. Calculate how frequently the pressure increases from one day to another vs. how frequently it decreases.
   4. Calculate the percentage of days when the humidity was above some value, like 90%.
   5. Calculate the mean and standard deviation of precipitation levels.

Be sure to include a descriptive sentence for what you are printing out in each case.

Our team’s interesting analysis questions 2 and 3:

* Average temperature in a January
* The total number of days with no rain
* The highest amount of rain in March (in inches)

Note that the “interesting” analysis questions you choose should be different analyses from each other; for example, you should not find the min/max temperature for just 3 different dates, or find the min/max pressure for December 25, since those are essentially the identical computation.

Code:

*# By submitting this assignment, I agree to the following:  
# “Aggies do not lie, cheat, or steal, or tolerate those who do”  
# “I have not given or received any unauthorized aid on this assignment”  
#  
# Name: ALEXIA PEREZ  
# Section: 508  
# Assignment: LAB 9B  
# Date: 28-10-2018***from** math **import \*  
import** numpy  
  
  
*# Assigning variables*file **=** open(**'WeatherDataMac.csv'**,**'r'**)  
maximum\_temp **=** 0  
minimum\_temp **=** 1**\***(10**\*\***27)  
total\_rain **=** 0  
count **=** 0  
total\_temp **=** 0  
count1 **=** 0  
count2 **=** 0  
height **=** 0  
  
*# Operations  
# Below each "try" there's a statement that contains its own function to get a desired value:***for** line **in** file**:** *# This next few lines will calculate the max and min temperatures:* **try:** t **=** float(line.split(**','**)[1])  
 **if** t **>** maximum\_temp**:** maximum\_temp **=** t  
 **except:  
 pass  
 try:** m **=** float(line.split(**','**)[3])  
 **if** m **<** minimum\_temp**:** minimum\_temp **=** m  
 **except:  
 pass** *# This next few lines will calculate the summation of the precipitation and number of days  
 # will later be used to calculate the average.* **try:** r **=** float(line.split(**','**)[13])  
 total\_rain **+=** r  
 count **+=** 1  
 **except:  
 pass** *# The next few lines will find the summation of all the average temperatures in January,  
 # and the total number of days.  
 # These 2 values will be later used to calculate the average temperature in January.* **try:  
 if** int(line[0].split(**'/'**)[0])**==**1**:** d **=** float(line.split(**','**)[2])  
 total\_temp **+=** d  
 count1 **+=** 1  
 **except:  
 pass** *# The next lines of code will find the highest precipitation in inches in a specific month (March):* **try:** q **=** float(line.split(**','**)[13])  
 **if** int(line[0].split(**'/'**)[0]) **==** 3**:  
 if** q**>** height**:** height **=** q  
 **except:  
 pass** *# The next few lines will find the days where there was 0 precipitation:* **try:  
 if** float(line.split(**','**)[13]) **==** 0**:** count2 **+=** 1  
 **except:  
 pass**average\_temp **=** total\_temp **/** count1  
average\_rain **=** total\_rain **/** count  
print(**'The maximum temperature was: '**, maximum\_temp,**". And the minimum temperature was: "**,minimum\_temp,**'**\n**'  
 'The average rain in inches was: '**,average\_rain,**'**\n**'  
 'The average temperature in January was: '**,average\_temp,**'**\n**'  
 'The highest amount of rain in March was: '**,height,**' inches.'**,**'**\n**'  
 'The total number of days with no rain was: '**,count2)

Output:

